

2/5/1 (Item 1 from file: 347)  
DIALOG(R) File 347:JAPIO  
(c) 2000 JPO & JAPIO. All rts. reserv.

04205886 \*\*Image available\*\*  
DEBUGGING SYSTEM FOR DISTRIBUTED INFORMATION PROCESSING SYSTEM

PUB. NO.: 05-197586 JP 5197586 A]  
PUBLISHED: August 06, 1993 (19930806)  
INVENTOR(s): KAWABE SHIGEHISA  
YAMASHITA ICHIRO  
APPLICANT(s): FUJI XEROX CO LTD [359761] (A Japanese Company or  
Corporation), JP (Japan)  
APPL. NO.: 03-060781 [JP 9160781]  
FILED: February 27, 1991 (19910227)  
INTL CLASS: [5] G06F-011/28; G06F-015/16; G06F-015/16  
JAPIO CLASS: 45.1 (INFORMATION PROCESSING -- Arithmetic Sequence Units);  
45.4 (INFORMATION PROCESSING -- Computer Applications)  
JOURNAL: Section: P, Section No. 1646, Vol. 17, No. 624, Pg. 13,  
November 17, 1993 (19931117)

#### ABSTRACT

PURPOSE: To easily execute debugging without being conscious of a remote procedure call in a distributed information processing system where plural processes make communication by means of the remote procedure call and realize a parallel operation.

CONSTITUTION: Plural satellite debugging means 3 and 4 which are connected to plural processes 1 and 2 and execute debugging, and a center debugging means 5 which remotely controls the operation of the plural satellite debugging means 3 and 4 are provided. The center debugging means 5 has a remote procedure call issue detecting means 51 which previously detects that there is the remote procedure call in the process of a debugging object in the satellite debugging means 3 (or 4), a remote procedure call end detecting means 52 detecting the end of the processing of the remote procedure call and a switch control means 53 which switches and controls the starting state of the satellite debugging means 3 and 4 in response to the detection of the remote procedure call issue detecting means 51 or the detection of the remote procedure call end detecting means 52.

(19)日本国特許庁(JP)

(12) 公開特許公報(A)

(11)特許出願公開番号

特開平5-197586

(43)公開日 平成5年(1993)8月6日

(51)Int.Cl. <sup>5</sup>	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 11/28	J	9290-5B		
15/16	3 7 0 N	9190-5L		
	4 5 0 Z	9190-5L		

審査請求 未請求 請求項の数1(全 8 頁)

(21)出願番号 特願平3-60781

(22)出願日 平成3年(1991)2月27日

(71)出願人 000005496

富士ゼロックス株式会社  
東京都港区赤坂三丁目3番5号

(72)発明者 川邊 恵久

神奈川県海老名市本郷2274番地 富士ゼロックス株式会社海老名事業所内

(72)発明者 山下 一郎

神奈川県海老名市本郷2274番地 富士ゼロックス株式会社海老名事業所内

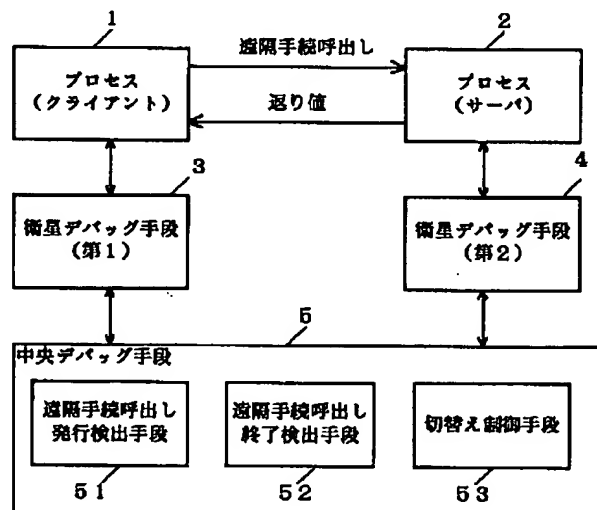
(74)代理人 弁理士 岩上 昇一 (外1名)

(54)【発明の名称】 分散型情報処理システムのデバッグ方式

(57)【要約】

【目的】 複数のプロセス1, 2が遠隔手続呼出しにより通信し、並行・並列動作を実現する分散型の情報処理システムにおいて遠隔手続呼出しを意識しないで簡単にデバッグができるようにすること。

【構成】 複数のプロセス1, 2の各々に接続してデバッグを行う複数の衛星デバッグ手段3, 4と、その複数の衛星デバッグ手段3, 4の動作を遠隔制御する中央デバッグ手段5とを備えている。そして、その中央デバッグ手段5は、衛星デバッグ手段3 (または4) のデバッグ対象のプロセスに遠隔手続呼出しがあることを前もって検出する遠隔手続呼出し発行検出手段51と、前記遠隔手続呼出しの処理の終了を検出する遠隔手続呼出し終了検出手段52と、遠隔手続呼出し発行検出手段51の検出あるいは遠隔手続呼出し終了検出手段52の検出に応じて、衛星デバッグ手段3, 4の起動状態の切り替え制御を行う切替え制御手段53とを有する。



## 【特許請求の範囲】

【請求項1】複数のプロセスが互いに通信を行い並行・並列動作を実現する分散型の情報処理システムのデバッグ方式において、前記複数のプロセス各々に接続してデバッグを行う複数の衛星デバッグ手段と、その複数の衛星デバッグ手段の動作を遠隔制御する中央デバッグ手段とを備え、その中央デバッグ手段は、衛星デバッグ手段のデバッグ対象のプロセスに遠隔手続き呼出しがあることを前もって検出する遠隔手続き呼出し発行検出手段と、前記遠隔手続き呼出しの処理の終了を検出する遠隔手続き呼出し終了検出手段と、前記遠隔手続き呼出し発行検出手段の検出あるいは前記遠隔手続き呼出し終了検出手段の検出にตอบสนองして、前記衛星デバッグ手段の動作状態の切り替え制御を行う制御手段と、を有することを特徴とする分散型情報処理システムのデバッグ方式。

## 【0001】

## 【発明の詳細な説明】

【産業上の利用分野】本発明は、複数の計算機を互いに関連制御することによって、複数のプログラムを分散して処理する分散型の情報処理システムに関し、特にこうしたプログラムの並行・並列あるいは分散処理に際してそのシステム内で並行・並列動作する各プロセスを対象として好適なデバッグ方式に関する。

## 【0002】

【従来の技術】近年、ネットワークの普及や計算機の低価格化と、計算機に要求される処理の複雑化と多様化により、複数のプロセス各々について通信チャンネルを介した通信手段を通じて互いに通信を行い、協調して並行・並列動作するプロセスからなるプログラムが記述されるようになってきている。特に、広域ネットワークや狭域ネットワークで結合された複数の計算機に分散配置された前記並行・並列動作するプロセスからなる分散プログラムのプログラミングが行われるようになってきている。このようなプログラミングにおいては、協調して並行・並列動作するプロセスを容易に記述する方法やデバッグする方法の開拓が大きな課題となっている。

【0003】協調して並行・並列動作するプロセスを容易に記述する方法の一つとしては、プログラムを、処理を提供するサーバと処理結果を利用するクライアントとに機能分離を行い、記述することによりモジュール化しプログラミングを行う方法がある。特に、サーバとクライアントの機能分離の方式の一つに遠隔手続き呼出しとして知られる方法がある（たとえば、「分散コンピューティング環境の最前線」、日経エレクトロニクス、1990、6、11、No. 502、pp122-148参照）。遠隔手続き呼出しにおいては、処理の単位を手続

きとして捉え、同一あるいは遠隔の計算機上のプロセスに対し手続きを指定し引数を送信し、手続きの戻り値を受信する。また手続きの副作用が情報処理システムの状態に反映されることで、処理が行われる。この遠隔手続き呼出しを導入することにより簡単に協調して並行・並列動作するプロセスからなるプログラムのプログラミングが可能である。デバッグに際しても簡単に協調して並行・並列動作するプロセスをいかに扱うかが大きな問題の一つとなっている。従来のデバッグ方式は単一のプロセスを扱うにすぎなかったが、近年では並行・並列動作するプロセスを扱うことが可能なデバッグ方式も提案されている（たとえば、(1) D.Caswell, D.Black 「Implementing a March Debugger For Multithreaded Applications」 Proc. of USENIX, Winter '90, Nov. 1989 pp 25-39、(2) 「Domain Distributed Debugging Environment Reference」 Apollo Computer Inc., Order No.011024-A00、(3) 特開平1-200563号公報等参照）。また、遠隔手続き呼出し機能に通信障害の発生や遠隔手続き呼出しのに関する再試行の情報を管理する機構を組み込み、遠隔手続き呼出しのデバッグを補助するデバッグ方式も提案されている（たとえば、R.Cooper 「Pilgrim: A Debugger for Distributed Systems」 Proc. of 7th Int Conf. on Distributed Computing Systems, pp.458-465, Sep. 1987参照）。

## 【0004】

【発明が解決しようとする課題】しかし、従来のプログラムデバッグ方式では、並行・並列動作するプロセスをデバッグ可能とする機能を重点として考えられており、特に遠隔手続き呼出しを用いた並行・並列動作するプロセスのプログラムをあたかも単一プロセスのプログラムと同様に簡単にデバッグすることは困難であるという不具合があった。本発明は、前記従来技術のプログラムデバッグ方式の持つ不具合に鑑み、遠隔手続き呼出しを用いて互いに通信を行う複数のプロセスにより実現される情報処理システムにおいて、遠隔手続き呼出しを意識しないで簡単にデバッグができるプログラムデバッグ方式を提供することを課題とする。

## 【0005】

【課題を解決するための手段】本発明は、上記課題を解決するために、図1に示すように、複数のプロセス1、2（図2の203または213で実行されるプロセス）が通信を行い並行・並列動作を実現する分散型の情報処理システムのデバッグ方式において、前記複数のプロセス1、2の各々に接続してデバッグを行う複数の衛星デバッグ手段3、4（図2の20、21）と、その複数の衛星デバッグ手段3、4の動作を遠隔制御する中央デバッグ手段5（図2の22）とを備えている。そして、その中央デバッグ手段5は、衛星デバッグ手段3（または4）のデバッグ対象のプロセスに遠隔手続き呼出しがあることを前もって検出する遠隔手続き呼出し発行検出手段

51(図2の224)と、前記遠隔手続呼出しの処理の終了を検出する遠隔手続呼出し終了検出手段52(図2の225)と、前記遠隔手続呼出し発行検出手段51の検出あるいは前記遠隔手続呼出し終了検出手段52の検出に応答して、前記衛星デバッグ手段3,4の起動状態の切り替え制御を行う切り替え制御手段53(図2の222)とを有する。

#### 【0006】

【作用】上記構成において、第1の衛星デバッグ手段3によりデバッグが行われているひとつのプロセス(以下、クライアントプロセス)1が遠隔手続呼出しを発行するとき、遠隔手続呼出し発行検出手段51はその遠隔手続呼出しの発行を検出するとともに、その要求される遠隔手続を含む他のプロセス(以下、サーバプロセス)2を特定する。中央デバッグ手段4は要求される遠隔手続を含むサーバプロセス2が動作している計算機上に第2の衛星デバッグ手段5を起動させる。また、切り替え制御手段53は、中央デバッグ手段5と衛星デバッグ手段との間の通信を、第1の衛星デバッグ手段3から第2の衛星デバッグ手段4に切り替えることにより、クライアントプロセスのデバッグを行う者はサーバプロセス2のデバッグをクライアントプロセス1のデバッグに引続き行う。上記遠隔手続の実行が終了すると、遠隔手続呼出し終了検出手段52は遠隔手続呼出しの実行の終了を検出する。切り替え制御手段53は、第2の衛星デバッグ手段4に、サーバプロセス2のデバッグを終了するように指示する。また、サーバプロセス2のデバッグが終了すると、第2の衛星デバッグ手段4の動作を終了させる。そして、第1の衛星デバッグ手段3にデバッグを行わせるよう起動状態を切り替える。これにより、サーバプロセス2のデバッグを行う者はサーバプロセス2のデバッグから引続きクライアントプロセスのデバッグを行うことができるものである。以上のように、本発明は、サーバプロセスとクライアントプロセスの衛星デバッグ手段を適切に制御することができるので、デバッグをする者には、遠隔手続呼出しが発行されたときの通信手順や引数の通信部分、あるいは、遠隔手続呼出しが終了したときの通信手順や戻り値の通信部分は見えなくて、あたかもクライアントプロセスから単なるシングルプロセス上の、遠隔手続ではない手続呼出しが実行され、戻り値が返されたかのように見える。従って、デバッグをする者は、従来の単一プロセスに対するデバッグ手法を用いることができ、プロセスの並行・並列動作を意識することなく簡単に遠隔手続呼出しを行う並行・並列分散プログラムのデバッグをすることができる。

#### 【0007】

【実施例】以下、図面を参照しながら実施例に基づいて、本発明の特徴を具体的に説明する。図2は、本発明の実施例を示すブロック図である。図2において、遠隔手続呼出し記述ファイルには、遠隔手続名と遠隔手続の

引数の型や数、戻り値の型が定義されている。たとえば図5において遠隔手続呼出し記述ファイルの例として、遠隔手続名がprintmessageで、引数はstringの型で1個を持ち、戻り値はint型を持つことが示されている。本実施例では、予めクライアントプロセス実行部203で実行されるクライアントプロセスには第1のソースコードデバッグ部206が結合されているものとする。しかしながら、クライアントプロセスが起動された後で、クライアントプロセスのプロセス番号を知り第1のソースコードデバッグ部206を接続してもよい。第1のソースコードデバッグ部206は通信部23における第1のTCPソケット231による通信チャネルを介して中央デバッグ部22のデバッガインタフェース222と結合している。この第1のTCPソケット231は、具体的には、第1の衛星デバッグ部20側に接続されたTCPソケットと、中央デバッグ部22側に接続されたTCPソケットと、それらのTCPソケット間を接続する通信線とからなっている。通信部23の他のソケットやアクセス部についても同様に通信線とその両端部のインタフェース部からなっている。

【0008】デバッグをする者は中央デバッグ部22のディスプレイ端末装置223からコマンドを発行し、そのコマンドはデバッガインタフェース222を通じて第1のソースコードデバッグ部206に送られる。第1のソースコードデバッグ部206は、クライアントソースファイル記憶部201のクライアントソースファイルを参照して、ソースコードのデバッグを行う。第1のソースコードデバッグ部206に与えられたコマンドの実行結果は、第1のTCPソケット231による通信チャネルを介してデバッガインタフェース222に送られディスプレイ端末装置223に表示される。

【0009】そこで例えばブレークポイントで停止しているクライアントプロセスに対し、ソースコードについて1ステップずつの実行を指示するステップ実行コマンドを発行すると、第1のソースコードデバッグ部206は、クライアントプロセス実行部203でのステップ実行をさせ、クライアントプロセスが再度停止すると、ソースコード中での現在の停止地点を第1のブレークポイント検出部224に出力する。

【0010】第1のブレークポイント検出部224は、第1のリモートファイルアクセス部232を介して、クライアントプロセスのソースコード中での現在の停止地点からクライアントソースファイル記憶部201のクライアントソースファイルを参照し、現在の停止地点が手続呼出しであるかどうかを調べ、手続呼出しでない場合は、デバッガインタフェース222には何も出力しない。また、現在の停止地点が手続呼出しであった場合は、遠隔手続呼出し記述ファイル記憶部226のファイルを参照し、手続呼出しの手続名と一致する遠隔手続名が存在すれば、現在の停止地点はクライアントプロセス

中で遠隔手続呼出しを行う行であるとみなし、デバッグインタフェース222に遠隔手続名と遠隔手続に用いられる通信チャンネルのチャンネル番号と遠隔手続が起動されることを通知する。通信チャンネルは、本実施例ではインタフェースとして第3のTCPソケット233が用いられ、例えば通信チャンネルのチャンネル番号としてはRFC1060(1990年3月、J.Reynolds J.Pstel ISI)に規定されているインターネットプロトコルのポート番号が挙げられる。図3のクライアントのソースコードにおいて、クライアントプロセスが行7で「result=printmessage(&message,cl);」の実行をする前に停止しているとすれば、第1のブレークポイント検出部224は、図5に例示する遠隔手続呼出し記述ファイル記憶部226のファイルにおける行cの「int PRINTMESSAGE(string)=1」と比較し、手続printmessageが遠隔手続PRINTMESSAGEに相当することを検出し、デバッグインタフェース222に遠隔手続printmessageが起動されることを通知する。デバッグインタフェース222はその第1のブレークポイント検出部224からの遠隔手続起動通知を記録する。

【0011】次に、デバッグをする者が再度ステップ実行を指示すると、デバッグインタフェース222は、第1のソースコードデバッグ部206に対して現在停止している「行8」の実行が終了したあと次に実行される行として「行9」に第1のブレークポイントを設定するよう通知する。また、デバッグインタフェース222は、記録された前記遠隔手続の起動通知の有無を調べ、通知の記録があれば、遠隔手続を含むプロセスを有する計算機名を調べる。この計算機名は、例えば、通知された遠隔手続名の呼出しに与えられる引数の中に前記遠隔手続を含むプロセスを有する計算機名が含まれているような遠隔手続の方式が用いられている場合は、第1のソースコードデバッグ部206にステップ実行を指示する代りに、手続呼出しの引数の詳細な内容を表示するコマンドを与え、その表示結果を調べることにより知ることができる。あるいは、遠隔手続の起動において、特定の計算機名を指定せずに遠隔手続投入先計算機決定部227によってその遠隔手続を含むプロセスを有する複数の計算機の中からひとつの計算機を決定するようにしてもよい。この場合は、デバッグインタフェース222はその遠隔手続投入先計算機決定部227からの出力により、遠隔手続を含むプロセスを有する計算機名を知ることができる。

【0012】デバッグインタフェース222は前記遠隔手続を含むプロセスを有する計算機上に第2のソースコードデバッグ部217を起動し、第2のTCPソケット234による通信チャンネルを介し、その起動した第2のソースコードデバッグ部217と接続する。次に、デバッグインタフェース222はプロセステーブルアクセス部236を用い、通知されている遠隔手続に用いられる

通信チャンネルのチャンネル番号から、前記遠隔手続を有し、クライアントプロセスからの接続を待っているプロセスのプロセス番号を特定し、第2のソースコードデバッグ部217に接続すべきサーバプロセスのプロセス番号として通知し、サーバプロセス実行部213と第2のソースコードデバッグ部217を接続する。デバッグインタフェース222は記録された前記遠隔手続の名前を第2のソースコードデバッグ部217に通知して、第2のブレークポイントを図4に例示するソースコードの「行ハ」に設定し、第3のブレークポイントを「行ホ」に設定する。

【0013】次に、第2のソースコードデバッグ部217に対して継続実行を指示するコンティニュー命令を通知することで、サーバプロセス実行部213はクライアントプロセス実行部203からの接続を待ち、第2のソースコードデバッグ部217は第2のブレークポイントに到達するまで継続実行する。その後、デバッグインタフェース222は、第1のソースコードデバッグ部206に対して継続実行を指示するコンティニュー命令を通知する。クライアントプロセスは遠隔手続の実行が終了し、第1のブレークポイントに到達するまで実行を続ける。その後、デバッグインタフェース222はディスプレイ端末装置223からの入力に主に第2のソースコードデバッグ部217に送られ、第2のソースコードデバッグ部217から出力された結果はディスプレイ端末装置223に出力されるように切り替えられる。

【0014】次に、遠隔手続がクライアントプロセス実行部203で実行中のプロセスから発行されると、サーバプロセス実行部213のサーバプロセスは第2のブレークポイントで停止する。このように、第1のソースコードデバッグ部206と第2のソースコードデバッグ部217を制御することで、デバッグをする者には、遠隔手続呼出しが発行されたときの通信手順や引数の通信部分は見えなくて、あたかもクライアントプロセスから単なるシングルプロセス上の遠隔手続ではない手続呼出しが実行されたかのように見える。デバッグする者は特にサーバプロセスのデバッグをしていることを意識せずに遠隔手続をデバッグできる。

【0015】遠隔手続のデバッグが終了し、例えばデバッグをする者がコンティニュー実行を指示すれば、デバッグインタフェース222はコンティニュー実行が指示されたことを記録し、サーバプロセス実行部213は「行ホ」において第3のブレークポイントで停止し、デバッグインタフェース222は第2のソースコードデバッグ部217から第2のTCPソケット234を通じてそれを知った後、第2のソースコードデバッグ部217に第2のブレークポイントと第3のブレークポイントを取り除くよう指示する。第2のブレークポイントと第3のブレークポイントが取り除かれたら、デバッグインタフェース222は、ディスプレイ端末装置223からの

コマンドが第1のソースコードデバッグ部206に送られるように切り替え、第1のソースコードデバッグ部206の出力がディスプレイ端末装置223に表示されるように切り替える。

【0016】その後、デバッグインタフェース222は第2のソースコードデバッグ部217にサーバプロセス実行部213のサーバプロセスのデバッグを終了するように指示する。その結果、サーバプロセス実行部213は第2のソースコードデバッグ部217との接続が切断され、継続実行される。サーバプロセス実行部213からの遠隔手続の返り値がクライアントプロセス実行部203に戻され、第1のブレークポイントでクライアントプロセスが一旦停止するが、ソースコードデバッグ部206は、デバッグをする者がサーバプロセスの遠隔手続においてコンティニュー実行を指示したことが記録されているため、第1のブレークポイントを取り除きコンティニュー実行を第1のソースコードデバッグ部206に指示する。このように第1のソースコードデバッグ部206と第2のソースコードデバッグ部217を制御することで、デバッグをする者には、遠隔手続呼出しが終了したときの通信手順や返り値の通信部分は見えなくて、あたかもシングルプロセス上の、遠隔手続ではない手続から返り値が戻されたかのように見える。従って、デバッグをする者は特にサーバプロセスのデバッグをしていることを意識せずに遠隔手続をデバッグすることができる。なお、以上に説明した実施例では、1つのクライアントプロセスと1つのサーバプロセス間の遠隔手続呼出しを例にとり本発明のデバッグ方式を説明したが、一般的には、一つのクライアントプロセスと並列に接続された複数のサーバプロセスからなる分散型の情報処理システムにおける任意のクライアントプロセスとサーバプロセス間のデバッグに本発明のデバッグ方式を適用できることは言うまでもない。また、クライアントプロセスにサーバプロセスが接続され、さらにそのサーバプロセスがクライアントプロセスとして別のサーバプロセスに接続されているような関係を含むシステムのデバッグにも本発明のデバッグ方式は同様に適用できる。

#### 【0017】

【発明の効果】以上述べたように、本発明によれば、サーバプロセスとクライアントプロセスの衛星デバッグ手段を適切に制御することにより、デバッグをする者に

は、遠隔手続呼出しが発行されたときの通信手順や引数の通信部分、あるいは、遠隔手続呼出しが終了したときの通信手順や返り値の通信部分は見えなくて、あたかもクライアントプロセスから単なるシングルプロセス上の、遠隔手続ではない手続呼出しが実行され、返り値が返されたかのように見える。従って、デバッグをする者は、従来の単一プロセスに対するデバッグ手法を用いることができ、プロセスの並行・並列動作を意識することなく簡単に遠隔手続呼出しを行う並行・並列分散プログラムのデバッグをすることができる。

#### 【図面の簡単な説明】

【図1】本発明の構成を示すブロック図、

【図2】本発明の実施例を示すブロック図、

【図3】遠隔手続呼出しを行うクライアントプロセスのソースコードの例を示す図

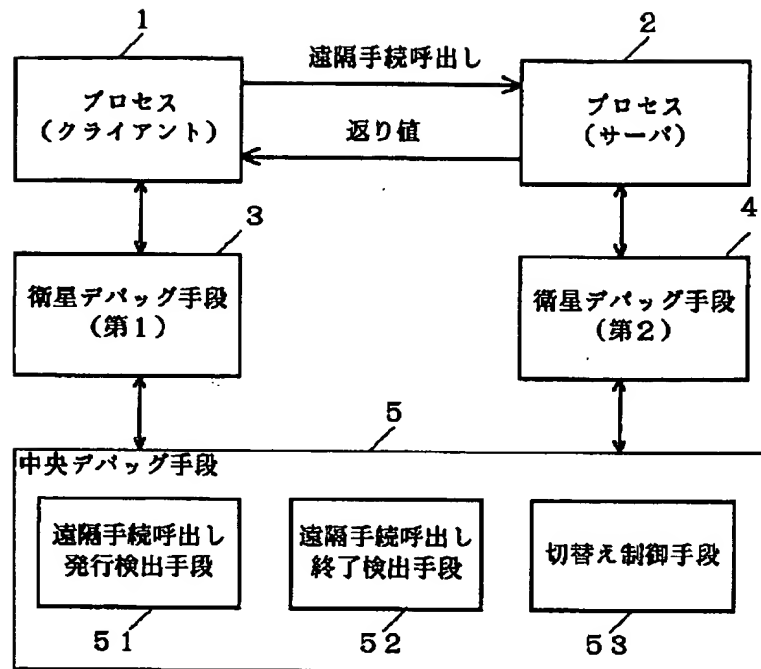
【図4】サーバプロセスのソースコードの例を示す図

【図5】遠隔手続呼出し記述ファイルの例を示す図。

#### 【符号の説明】

1、2…プロセス、3、4…衛星デバッグ手段、5…中央デバッグ手段、51…遠隔手続呼出し発行検出手段、52…遠隔手続呼出し終了検出手段、53…切替え制御手段、20…第1の衛星デバッグ部、201…クライアントソースファイル記憶部、202…クライアントプロセス実行形式ファイル記憶部、203…クライアントプロセス実行部、204…データ記憶部、206…第1のソースコードデバッグ部、21…第2の衛星デバッグ部、211…サーバソースファイル記憶部、212…サーバプロセス実行形式ファイル記憶部、213…サーバプロセス実行部、214…データ記憶部、217…第2のソースコードデバッグ部、218…プロセステーブル218、22…中央デバッグ部、222…デバッグインタフェース、223…ディスプレイ端末装置、224…第1のブレークポイント検出部、225…第2のブレークポイント検出部、226…遠隔手続呼出し記述ファイル記憶部、227…遠隔手続投入先計算機決定部227、23…通信部、231…第1のTCPソケット、232…第1のリモートファイルアクセス部、233…第3のTCPソケット、234…第2のTCPソケット、235…第2のリモートファイルアクセス部、236…プロセステーブルアクセス部236。

【図1】



【図3】

```

1 main(server,message){
2
3   cl=clnt_create(server,MESSAGEPROG,MESSAGEVERS);
4   if(cl==NULL){
5     exit(1);
6   }
7   result=printmessage(&message,cl);
8 }

```

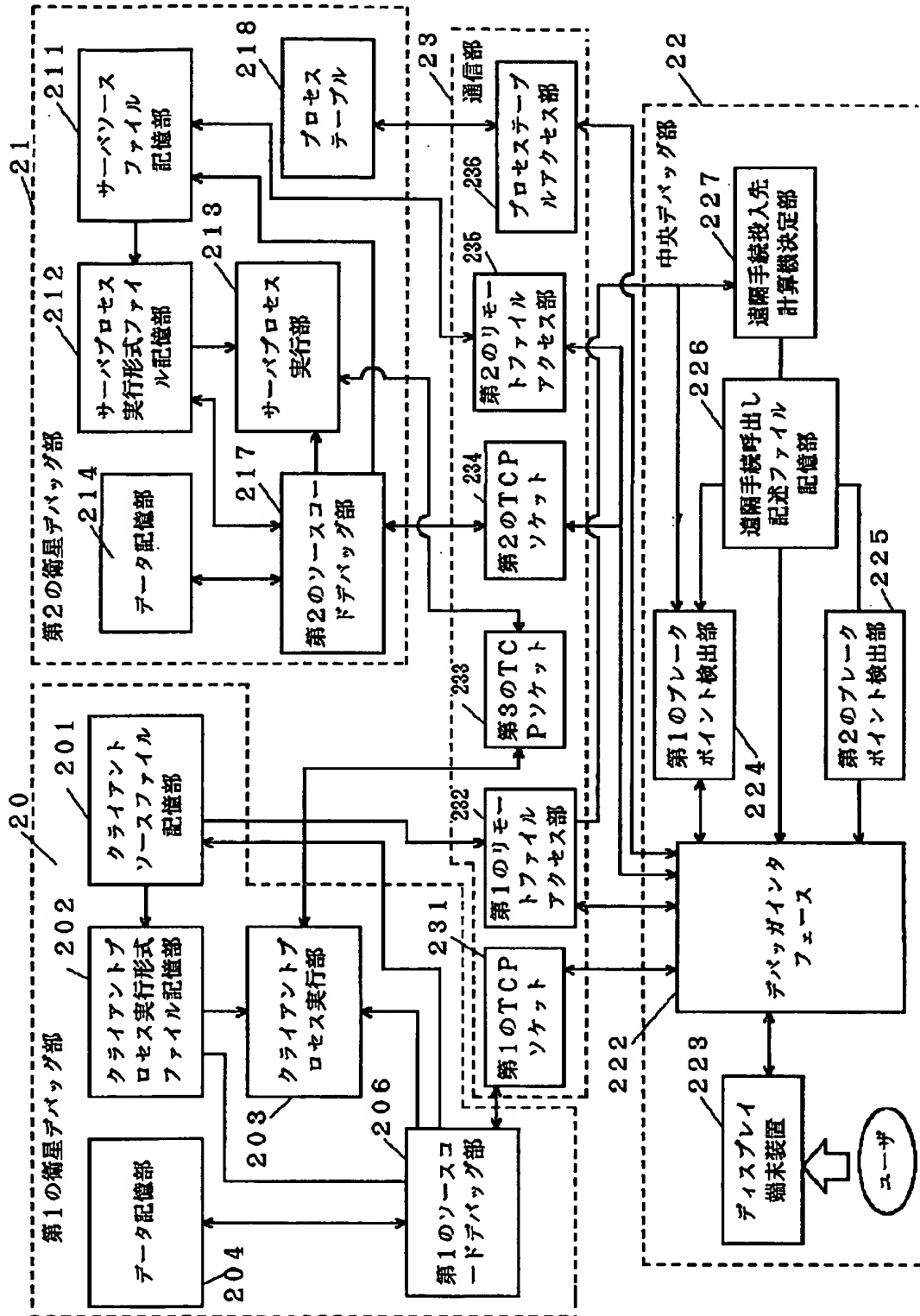
【図4】

```

1 printmessage(message){
2
3   printf("%s\n",*message);
4   result=1;
5   return(&result);
6 }

```

【図2】





【図5】

```
a program MESSAGEPROG{  
b   version MESSAGEVERS{  
c     int PRINTMESSAGE(string)=1;  
d   }=1;  
e }=99;
```